Searching Heuristically Optimal Path Using a New Technique of Bug0 Algorithm in Swarm Robotics

Hamza Sohail[1*] *, Muhammad Haris Yazdani*[2], *Rana Talal Ahmad Khan*[2]

[1]Department of Mechatronics Engineering, National University of Sciences and Technology (NUST), CEME, 43701 Rawalpindi, Pakistan
[2] Department of Mechanical Engineering, National University of Sciences and Technology (NUST), CEME, 43701 Rawalpindi, Pakistan

Abstract. Bug Algorithms in robotics field play an important role in path planning. The main challenge in conventional bug algorithms is searching the cluttered environment. To solve this problem a method is introduced which uses the concept of swarm robotics that helps in finding path using coordination between robots in swarm. The challenge in this research article is to find a path which is heuristically optimal. A type of bug algorithm is introduced in which parent bug sends two of its child bugs. Each of them has capability of searching in different directions. After searching the path from both sides, parent bug follows the path which is heuristically optimal. Parent and child bugs are equipped with tactile sensors to follow the perimeter of an obstacle. Illustrative simulation results show two test cases in which different scenarios are presented. Results are compared with of bug0 algorithm that is visualized in configuration space as well as in workspace to find the heuristically optimal path.

Keywords. *Mobile Robotics; Swarm Robotics; Heuristics; Path Planning; Bug Algorithm; Motion Planning*

1 Introduction

Traditional bug algorithms work on the principle of wall following. It is a path planning technique that was first originated from maze solving techniques. Such algorithms are used in indoor navigation due to their limited memory [1]. Bug algorithms were evolved from A* and dijkstra algorithm techniques used in late 80s [2, 3]. Different types of bug algorithms exist out of which three type of algorithms are traditional [4] that are bug0, bug1 and bug2. Bug0 is a simple algorithm which is also known as greedy algorithm in which bug follows the boundary as long as free space is encountered. It starts moving towards the goal as soon as it is encountered in a free space [5]. Bug0 is simply memory less algorithm. Bug1 is an advanced algorithm as compared to bug0 that stores the memory while following the perimeter of an obstacle. It is also considered as exhaustive search bug as it takes time to search an optimal path. Advancing further a new type of bug was introduced which was bug2. It follows the boundary as long as it gets to the M-line. M-line is the distance line from start to goal position. These bugs sense the obstacle using tactile sensors. Range sensor bug also known as tangent bug [6, 7] works on the principle of heuristics.

The problem that is faced in bug algorithms is that it can follow the perimeter of obstacle either in left or right direction. Swarm robotics help in solving such case [8]. Swarms robots are a group of robots that work in coordination such that each swarm robot is connected

* Hamza Sohail: hamza.sohail@ceme.nust.edu.pk

with the other swarm. Applications include search and rescue operations where a large robot cannot enter. Such issue can be resolved by path generation using swarm bug algorithms. Such swarm bugs are sent in these environments having low memory storage. Path planning in swarm robotics can also be useful in path generation using SLAM techniques [9].

Keeping in mind all the above cases, a new type of bug algorithm in swarm robotics is introduced. It consists of a parent bug which can eject two of its child bugs, which can follow the perimeter of obstacle in alternate directions. One child bug is programmed to move in left direction and second one in right direction of an obstacle. Even if one bug is lost in the loop the second one can reach the goal position and parent bug can trace the path of such child bug to reach the goal. Parent and child bugs are in coordination with each other due to their swarm characteristics. Results include illustrative simulation swarm bug algorithm. It shows how heuristically optimal path is reached by using the algorithm as compared with traditional bug0 algorithm.

2 Methodology

A circular robot equipped with tactile sensor is considered which can move in a workspace. Robot has the capability of ejecting multiple child bugs. This robot is named as parent bug. Functionality of each child bug is different. One of them has the capability of moving left direction and the other has the capability of moving in right direction. Child bug also has the capability of sending a signal to parent bug using wireless transmission after which

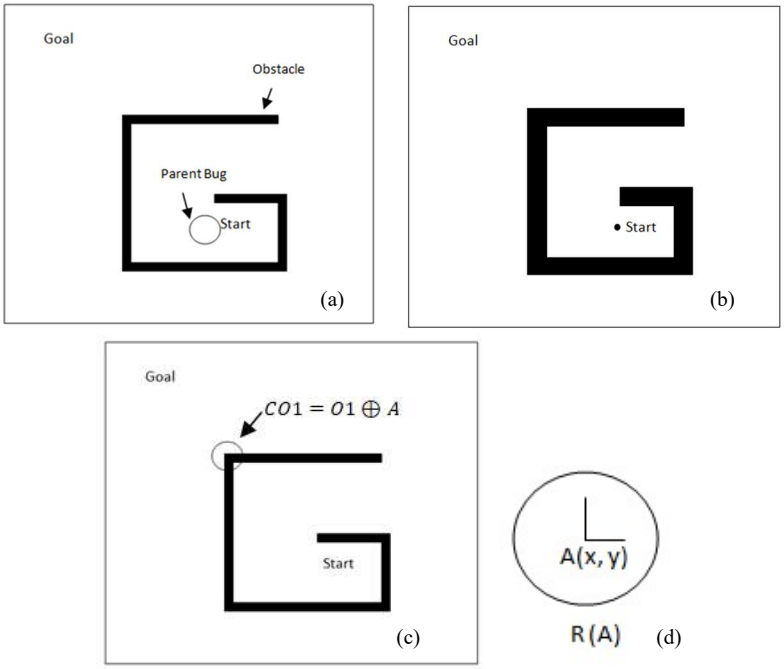

Figure 1 (a) Workspace including obstacles and robot, (b) Configuration Space, (c) Minkowski's Sum (d) Robot Shape

parent bug follows such path. If the path is not available then parent bug will stop as there is no path to move. Workspace defined for bug to move is $W \in \mathbb{R}^2$. Since visualization is in 2D space we can only take XY plane. Parent bug denoted as R (A) and child bugs as R(A1) and R(A2) can be represented in equation 1.

$$R(A) = A(X,Y) \in W \quad (1)$$

Same case for R(A1), R(A2), R(A3), and so on. Free workspace is defined as the whole workspace subtracted from obstacles and bugs. This also includes child bugs.

$$Wfree = W - \sum_{i=1}^{n} Oi - \sum_{j=0}^{k} R(Aj) \quad (2)$$

'k' in equation 2 represents number of child bugs R(A) ejects where R(A0) is the parent bug such that R(A0) = R(A). *Oi* represent number of obstacles where *Wfree* is the free workspace in which bug can move. Workspace leaves us with a major concern if we increase size of our R(A) than path might not be found. In this case configuration space visualization helps us in enhancing the size of obstacles and decreasing the size of robot to a point. Now 'A' is a center point of parent bug in configuration space which is shown in figure 1(d).

Figure 1(a) shows workspace in which parent bug and obstacles are included. Boundary of workspace is considered open. Figure 1(b) shows Configuration space [10] in which algorithm will be tested. Obstacles in configuration space are enlarged so that parent bug can be considered as a point robot. Child bugs are smaller than parent bugs. Child bugs are placed inside the parent robot. As soon as parent bug senses the obstacle using tactile sensor, both of the child bugs get ejected and follows the perimeter of the obstacle's boundary. There is no need to specifically enlarging for each swarm child bugs in configuration space. Configuration space is made by using minkowski's sum [11] in which center of parent bug is placed at boundary of obstacle, resulting in enlarging the obstacle such that reducing the size of parent bug to a point. Illustration is shown in Figure (c). The equation 3 shows CO1 as obstacle in configuration space.

$$CO1 = O1 \oplus R(A) \quad (3)$$

Tactile sensor works when parent bug collides with the obstacle which is defined as $R(A) \cap CO1$, it will stop and ejects two of its child bugs R(A1) and R(A2), both will follow the boundary in opposite directions.

Figure 2 shows the algorithm that starts with parent bug moving towards the goal, if no obstacle detected then it has been reached to goal. If obstacle is detected then it will eject two of its child bugs R(A1) and R(A2). R(A1) will move in left direction and R(A2) will move in right direction. R(A1) and R(A2) follows the boundary as long as it can move directly towards the goal. The time required by R(A1) is t1 while for R(A2) is t2. If t1<t2 then parent bug will follow R(A1) and vice versa. If none of them can reach the goal then no path. If one of them is lost in the loop then that child bug should lose connection from parent bug.

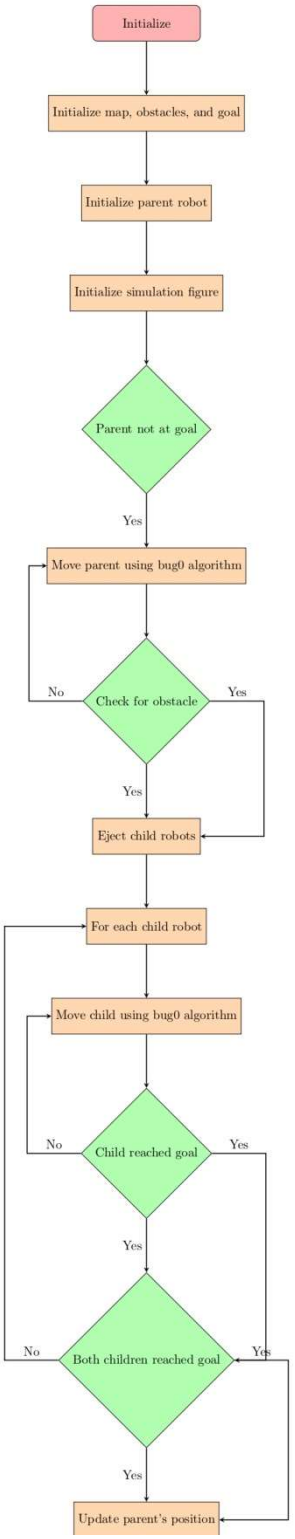

Figure 2 Flowchart

3 Results and Discussions

Proposed method suggests that searching technique used in this research article is a greedy solution which can work in certain conditions. Such an algorithm can be called as resolution complete algorithm.

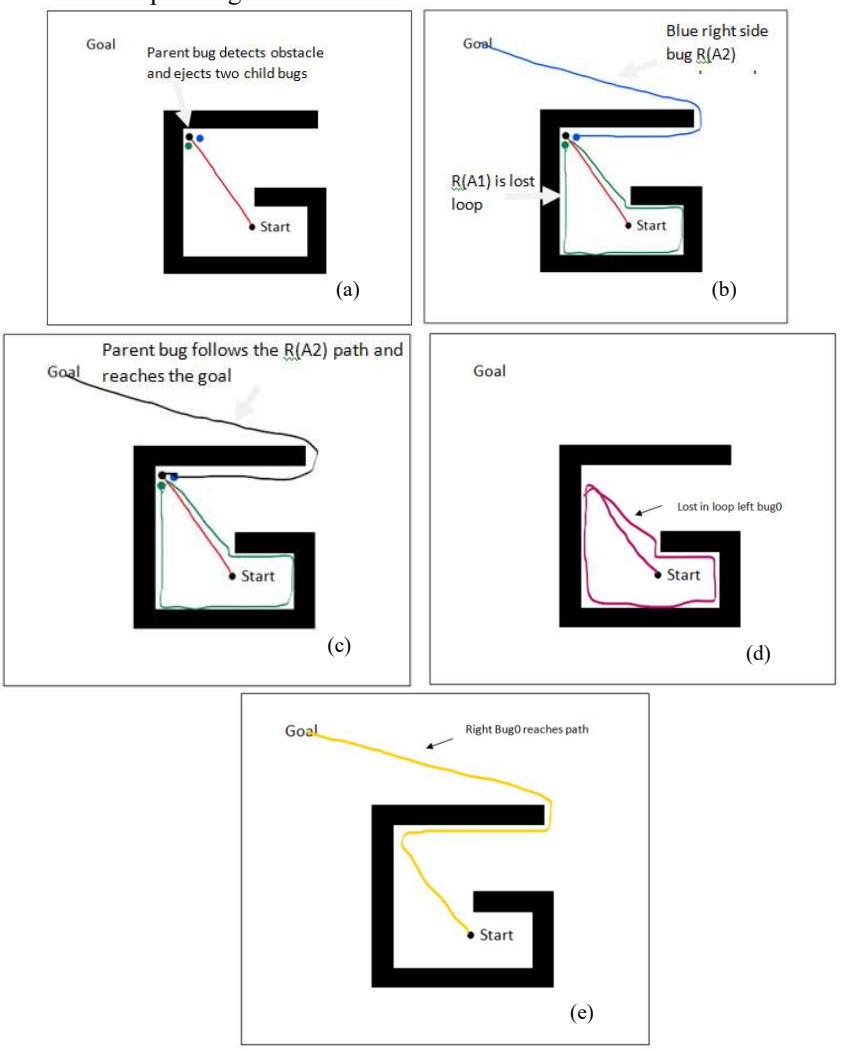

Figure 3 (a) parent bug obstacle detection, (b) Wall following of R(A1) & R(A2), (c) Parent bug decision, (d) Left Bug0, (e) Right Bug0

Given test case proposed in methodology section shows start and goal position of a parent bug which is visualized in configuration space to avoid size limitations of a bug. Figure 3 shows illustrative simulation of bug in configuration space.

Like traditional bug algorithms, the proposed robot is equipped with a tactile sensor that measures force when interacting with the physical body. Figure 3(a) shows movement of parent bug towards goal. As soon as it collides with obstacle, it ejects two of its child bugs. Both R(A1) and R(A2) starts following wall in their direction as shown in figure 3(b). It shows how R(A1) is lost in the loop while R(A2) reaches the path. On reaching the path

R(A2) sends a signal to parent bug such that R(A1) is moved back towards parent bug and they start following path of R(A2) as shown in figure 3(c). This shows how it is different than bug0 as it has the capability of searching left and right direction simultaneously. This test case only represented if one of its child bug is lost in the loop. There should be another test case in which both child bugs reach the goal.
As we can see in figure 4(a) the test case is simple in which both child bugs can reach goal. Blue child bug reaches the goal in 10 seconds while green child bug can reach the goal in 15 seconds as shown in figure 4(b). Parent bug needs a heuristic approach t2<t1 so it will follow the path as provided by the blue child bug as shown in figure 4(c). In above case total time taken by the parent bug to reach the goal is 2 times taken by the blue child bug to reach the goal. These 2 test cases confirm that this approach is better than bug0. Since bug0 can either take left or right direction in wall following. If left side was taken by the bug0 in figure 3(a) than it will be trapped inside a loop. Such case would fail if the bug0 was programmed only left. If programmed right bug0 than there will be no problem in reaching the goal. But a robot in an unidentified environment will not be able to judge whether to move left or right. Such case is resolved by swarm bug which searches the whole workspace by ejecting its child bugs and moves them in right and left direction.

Figure 3(d) shows left bug0 which is lost in path, it cannot reach path. In such case new algorithm is also time optimal but in case of right bug0 which reaches the goal in 10 seconds is more time optimal than the algorithm proposed as shown in figure 3(e).
Above algorithm shows how this parent bug acts like bug0 but it has the capability of searching the obstacle boundary from both sides. If above algorithm is changed to tangent like bug for each child bug it can be assumed that this is a resolution complete algorithm. Resolution complete algorithm means that the algorithm is not complete in all cases such that if both child bugs are lost in the loop. Comparison of parent bug with bug0 in test case 1 is given in table 1.

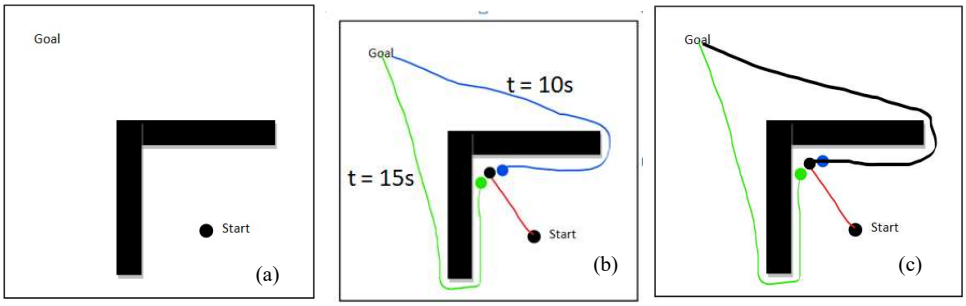

Figure 4(a) Test Case 2, (b) R(A1) and R(A2) path, (c) Parent bug path

Table 1 Comparison between Left, right Bug0 and Parent Bug in test case 1

Left Bug0	Right Bug0	Parent Bug
No path	Reach Goal in 10 seconds	Reach Goal in 20 seconds
Not complete Algorithm	Not Complete Algorithm	Resolution Complete Algorithm
Greedy Algorithm	Greedy Algorithm	Greedy Algorithm

In figure 5 the results are simulated in MATLAB it shows a parent bug, as soon as it hits an obstacle child bug 1 and child bug2 ejects and reach the goal. The problem in simulation is

that it runs in sequence so it is shown in two different figures. Parent bug will follow the path in figure 5(b) as the distance is less as compared to child bug 1 which is shown in figure 5 (a).

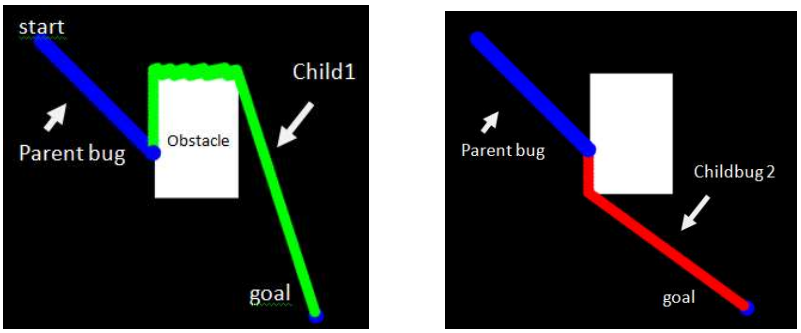

Figure 5 Simulation in MATLAB (a) child bug1 (b) child bug2

4 Limitations

Figure 3 (d) represents child bug movement in left direction in which it gets lost inside a loop within the boundary of an obstacle. This is also known as local minima. Proposed method allows the parent bug to wait until both right and left child bug searches the workspace. Even if one child bug falls in local minima, the other could reach its goal and allows parent bug to follow such path.

Figure 6 shows simulation in two more test cases in which one test case represents local minima. It shows how left bug tries to move towards goal position but gets stuck in the loop encountering corner of the obstacle. Figure 6 (b) shows successful test case in which path for both child bug exists. Left bug wins over right child bug in this test case.

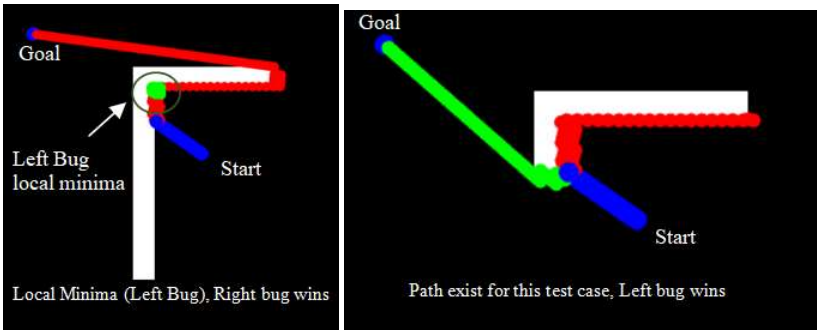

Figure 6 Simulation (a) Local Minima (b) Successful test case

In dynamic environments, such algorithm does not work using tactile sensor due to physical limitations. Since tactile sensor needs it to be collided, it may cause greater impact in dynamic environment if moving obstacle is at very high speed. In this case robot equipped with LIDAR or Sonar would be able to detect the obstacle and change its trajectory like tangent bugs.

5 Conclusions

We proposed a new type of bug algorithm in swarm robotics in which child bugs follows the perimeter of an obstacle from right and left direction to search heuristically optimal path. One that reaches the goal first sends a signal to parent bug which follows the path to reach goal. Two test cases are presented in the research paper. Test case 1 shows how one child bug is lost in the loop while the second reaches the path while in test case 2 both child bugs reach the path. In this case parent bug will follow the path of child bug that has reached first to the goal. In the end results are compared with bug0 algorithm which shows how our algorithm is better in certain conditions than bug0 algorithm. Weakness of proposed algorithm is that it is a greedy algorithm like bug0 but it is complete in certain conditions as it follows the perimeter of obstacle from both directions. Even if the path is restricted from one side, parent bug can reach goal from the other direction. In future the algorithm can be improved for other type of bug algorithms.

References:

1. McGuire, Kimberly N., Guido CHE de Croon, and Karl Tuyls. "A comparative study of bug algorithms for robot navigation." *Robotics and Autonomous Systems* 121 (2019): 103261.

2. P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, July 1968, doi: 10.1109/TSSC.1968.300136.

3. E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1) (1959) 269–271, URL https://doi.org/10.1007/BF01386390.

4. J. Ng and T. Bräunl, "Performance Comparison of Bug Navigation Algorithms", *2007 J Intell Robot Syst*, vol. 50, pp. 73-84.

5. S. S, D. A. Nandesh, R. Raman K, G. K and R. R, "An Investigation of Bug Algorithms for Mobile Robot Navigation and Obstacle Avoidance in Two-Dimensional Unknown Static Environments," 2021 International Conference on Communication information and Computing Technology (ICCICT), Mumbai, India, 2021, pp. 1-6, doi: 10.1109/ICCICT50803.2021.9510118.

6. I. Kamon, E. Rivlin and E. Rimon, "A new range-sensor based globally convergent navigation algorithm for mobile robots", *1996 Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 429-435.

7. A. Sankaranarayanan and M. Vidyasagar, "A new path planning algorithm for moving a point object amidst unknown obstacles in a plane", *1990 Proceedings. IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1930-1936.

8. Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Bram-billa, Arne Brutschy, et al . 2013. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. IEEE Robotics & Automation Magazine 20, 4 (2013), 60–71. https://doi.org/10.1109/MRA.2013.2252996

9. Johansson, Alexander, and Johan Markdahl. "Swarm Bug Algorithms for Path Generation in Unknown Environments." *arXiv preprint arXiv:2308.07736* (2023).

10. R. Pepy and A. Lambert, "Safe Path Planning in an Uncertain-Configuration Space using RRT," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, 2006, pp. 5376-5381, doi: 10.1109/IROS.2006.282101.

11. Agarwal, Pankaj K., Eyal Flato, and Dan Halperin. "Polygon decomposition for efficient construction of Minkowski sums." *Computational Geometry* 21.1-2 (2002): 39-61.